Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Studenti: Bilalović Meris, SV01/2023

Nikolić Dalibor, SV13/2023

Predmet: Nelinearno programiranje i evolutivni algoritmi

Broj projektnog zadatka: 2

Tema projektnog zadatka: Genetski algoritam, problem ranca 0/1

## **1. Uvod**

Problem ranca (0/1 Knapsack problem) je NP-teški kombinatorni optimizacioni problem u kome je potrebno odabrati podskup predmeta koji se mogu staviti u ranac, tako da se maksimizira ukupna vrednost, dok ukupna težina ne prelazi zadati kapacitet. Svaki predmet se može izabrati najviše jednom.

Zbog njegove izražene kompleksnosti (eksponencijalnog rasta broja kombinacija), koristi se heuristički pristup - genetski algoritam (GA), koji je inspirisan prirodnom evolucijom i procesima selekcije, ukrštanja i mutacije.

Cilj ove dokumentacije je da detaljno prikaže implementaciju, eksperimentalne rezultate i donete odluke u vezi sa konfiguracijom algoritma.

## **2. Opis ulaznih podataka**

Ulazni fajl sadrži podatke o kapacitetu ranca i listi predmeta. Prvi red sadrži kapacitet ranca, dok svaki sledeći red sadrži dva broja: težinu i vrednost jednog predmeta.

**Primer ulaznih podataka:**

capacity: 6404180  
382745,825594  
799601,1677009  
909247,1676628  
729069,1523970  
467902,943972  
... (još 18 predmeta)

Ukupno: 24 predmeta.

## **3. Struktura programa**

Program je napisan u Python programskom jeziku. Osnovne komponente:

### **3.1. Učitavanje podataka**

def load\_knapsack\_data(filepath):  
 with open(filepath, 'r') as f:  
 lines = f.readlines()  
 capacity = int(lines[0].split(':')[1].strip())  
 items = []  
 for line in lines[1:]:  
 weight, value = map(int, line.strip().split(','))  
 items.append((weight, value))  
 return capacity, items

*Funkcija vraća kapacitet i listu (težina, vrednost) parova.*

### **3.2. Reprezentacija rešenja**

Jedinka (hromozom) je predstavljena kao binarni niz dužine n (broj predmeta). Bit 1 označava da je predmet izabran.

### **3.3. Funkcija prilagođenosti**

def knapsack\_cost(chromosome):  
 total\_weight = 0  
 total\_value = 0  
 for gene, (weight, value) in zip(chromosome, items):  
 if gene == 1:  
 total\_weight += weight  
 total\_value += value  
 if total\_weight > capacity:  
 return 0  
 return -total\_value # jer rangiranje vrši minimizaciju

*Važno: penalizacija preteških rešenja vraćanjem nule.*

## **4. Selekcija, ukrštanje i mutacija**

### **4.1 Selekcija**

Koristili smo **ruletski metod selekcije** jer omogućava stohastički izbor jedinki prema vrednosti, uz zadržavanje diverziteta. Pored toga, koristi se i **prirodna selekcija** za elitizam.

### **4.2 Ukrštanje**

Eksperimentisali smo sa:

* **Jednotačkastim ukrštanjem** (finalna odluka zbog jednostavnosti i dovoljno dobrih rezultata)
* **Dvotačkastim ukrštanjem** (kompleksniji potomci, ali bez značajne razlike u kvalitetu)

Primer implementacije:

def one\_point\_crossover(pairs):  
 length = len(pairs[0])  
 children = []  
 for (a,b) in pairs:   
 r = random.randrange(0, length)  
 children.append(a[:r] + b[r:])  
 children.append(b[:r] + a[r:])  
 return children

### **4.3 Mutacija**

**Bit-flip mutacija** (dominantno korišćena): jednostavna i efikasna

**Inverzija:** testirana ali pokazala slabije rezultate u ovom tipu problema

## **5. Parametri algoritma**

|  |  |
| --- | --- |
| **Parametar** | **Vrednost** |
| Populacija | 100 jedinki |
| Generacije | 100 |
| Stopa mutacije | 0.1 |
| Elitizam | 10% |
| Broj roditelja | 50 |
| Tip selekcije | Ruletska |
| Ukrštanje | Jednotačkasto |

Ove vrednosti su eksperimentalno odabrane kao stabilne za dati skup podataka.

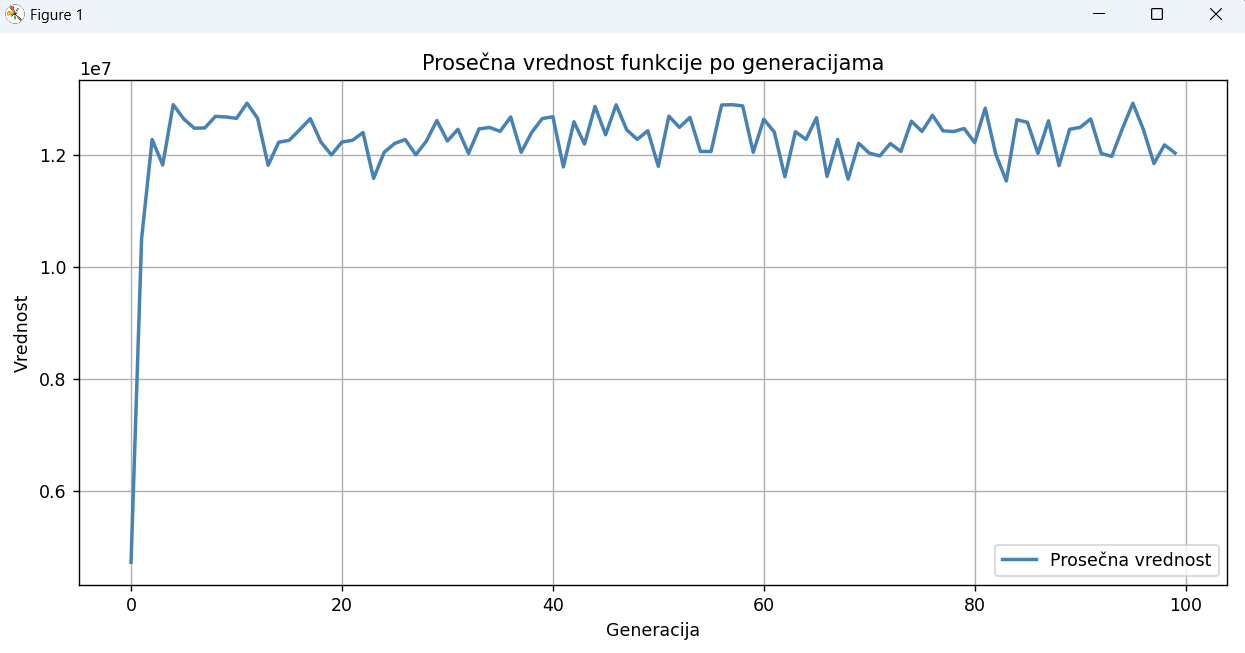
## **6. Rezultati algoritma**

Na osnovu pokretanja algoritma dobijeno je sledeće:

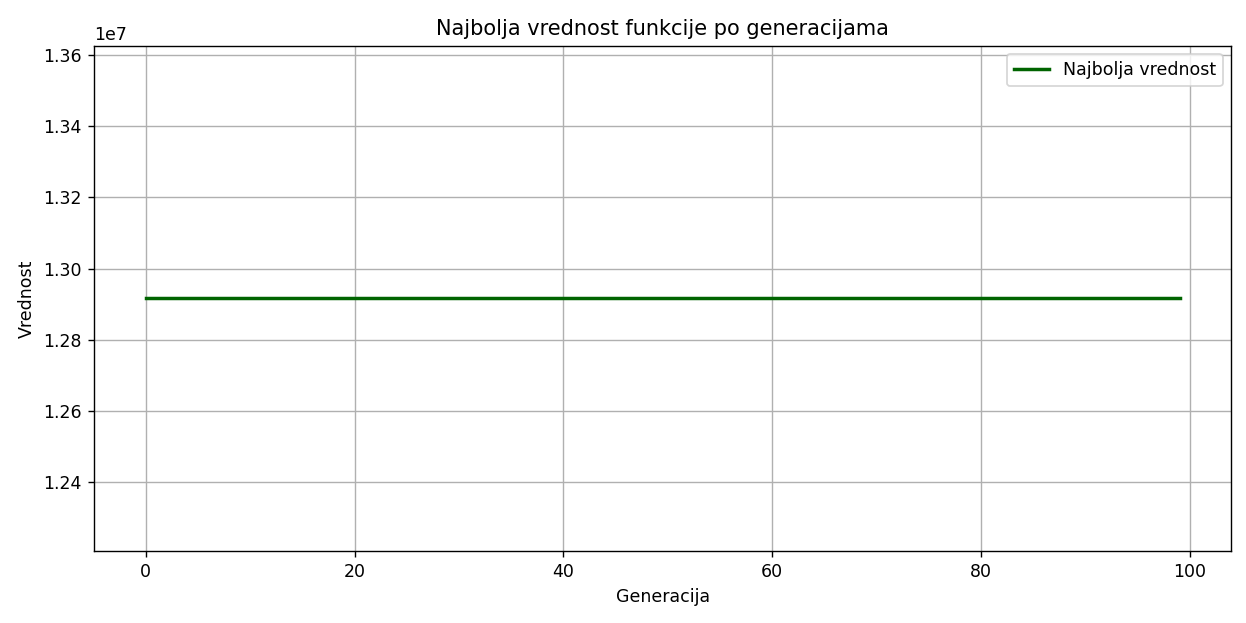
Najbolje rešenje: [1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1]  
Ukupna vrednost: 13202970  
Ukupna težina: 6390438

Grafik prikazuje konvergenciju tokom 100 generacija:

* **Prosečna vrednost funkcije po generacijama**



* **Najbolja vrednost funkcije po generacijama**



## **8. Zaključak**

Genetski algoritam se pokazao kao uspešna metoda za 0/1 problem ranca. Eksperimenti su pokazali da korišćenje ruletske selekcije i jednotačkastog ukrštanja dovodi do stabilne konvergencije. Mutacija se pokazala kao ključna za održavanje diverziteta.

Dalja poboljšanja mogu uključiti:

* Prilagodljivu mutaciju
* Lokalno poboljšavanje rešenja (hill climbing)
* Višekriterijumsko rangiranje

## **9. Literatura**

Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, 1989.

Predavanja sa kursa "Nelinearno programiranje i evolutivni algoritmi"

<https://en.wikipedia.org/wiki/Knapsack_problem>